# Assignment -4

PROJECT NAME	Digital Naturalist – AI Enabled Tool for Biodiversity Researchers
NAME	Azhagu Anu K
ROLL NO	950919106001
TEAM ID	PNT2022TMID49968

## 1. Import the necessary libraries

import pandas as pdimport numpy
as np
import matplotlib.pyplot as pltimport
seaborn as sns
from sklearn.model\_selection import
train\_test\_splitfrom sklearn.preprocessing import
LabelEncoder from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input, Embeddingfrom
keras.optimizers import RMSprop
from keras.preprocessing.text import
Tokenizerfrom keras.preprocessing import
sequence from keras.utils import pad\_sequences
from keras.utils import to\_categorical from
keras.callbacks import EarlyStopping

#### 2. Read dataset and do pre-processing

#### (i) Read dataset

2

df = pd.read\_csv('/content/spam.csv',delimiter=',',encoding='latin-1')
df.head()

**Unnamed: Unnamed: Unnamed:** 

v1 v22 3 4

3 4

0 ham Go until jurong point, crazy.. Available only ... NaN NaN 1 ham Ok lar...Joking wif u oni... NaN NaN NaN

NaN 2 spamFree entry in 2 a wkly comp to win FA Cup

3 ham U dun say so early hor... U c already then say... NaN NaN NaN 4 ham Nah I don't think he goes to usf, he lives aro... NaN NaN NaN



#### (ii) Preprocessing the dataset

```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True) df.info()
       <class 'pandas.core.frame.DataFrame'>RangeIndex:
       5572 entries, 0 to 5571
       Data columns (total 2 columns):
        # Column Non-Null Count Dtype
              v1 5572 non-null object
              v2 5572 non-null object dtypes: object(2) memory usage:
         1
       87.2+ KB
X = df.v2
Y = df.v1le = LabelEncoder() Y =
  le.fit transform(Y)
Y = Y.reshape(-1,1)
X train, X test, Y train, Y test = train test split(X, Y, test size=0.15)
max words = 1000
max len = 150
tok = Tokenizer(num words=max words) tok.fit on texts(X train)
sequences = tok.texts_to_sequences(X_train)
sequences matrix = pad sequences(sequences,maxlen=max len)
3,4. Create model and Add Layers(LSTM, Dense-(Hidden Layers), Output)
inputs = Input(name='inputs',shape=[max len])
layer = Embedding(max words,50,input length=max len)(inputs)
layer = LSTM(64)(layer) layer = Dense(256,name='FC1')(layer)
layer = Activation('relu')(layer) layer = Dropout(0.5)(layer)
layer = Dense(1,name='out layer')(layer) layer =
Activation('sigmoid')(layer) model =
Model(inputs=inputs,outputs=layer)model.summary()
       Model: "model"
        Layer (type) Output Shape Param #
```

```
= inputs (InputLayer) [(None, 150)] 0

embedding (Embedding) (None, 150, 50) 50000

lstm (LSTM) (None, 64) 29440 FC1 (Dense)

(None, 256) 16640 activation (Activation)

(None, 256) 0 dropout (Dropout) (None, 256) 0

out_layer (Dense) (None, 1) 257 activation_1

(Activation) (None, 1) 0

= Total params: 96,337

Trainable params: 96,337

Non-trainable params: 0 5.
```

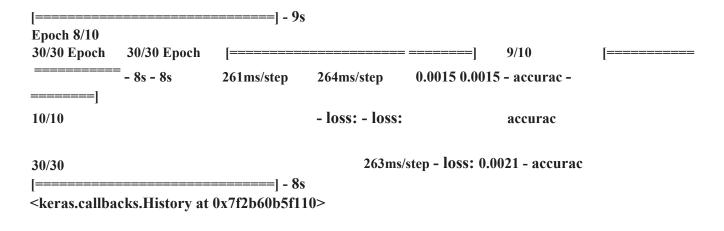
## Compile the model

model.compile(loss='binary\_crossentropy',optimizer=RMSprop(),metrics=['accuracy']) 7. Train

### and Fit the model

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10, validation_split=0.2)
```

```
Epoch 1/10
30/30 [==
                                         ==| - 8s 263ms/step - loss: 0.0060 - accurac
Epoch 30/30 Epoch 30/30
                                                                   =======| - 8s
2/10
                                                 263ms/step - loss: 0.0572 - accurac
263ms/step - loss: 0.0036 - accurac 3/10 Epoch
4/10
30/30 Epoch
                                ====] 5/10
                                                                 accurac
                           === - 8s 262ms/step - loss: 0.0038 -
                            263ms/step
                                                                       0.0018 0.0022 accurac
30/30 Epoch
             6/10
30/30 Epoch
                            7/10
                                          261ms/step
                                                        - loss: - loss: - accurac -
30/30
                                                 310ms/step - loss: 0.0020 - accurac
```



## 6. Save the model

model.save('sms\_classifier.h5')

Preprocessing the Test Dataset

test\_sequences = tok.texts\_to\_sequences(X\_test)
test\_sequences\_matrix = pad\_sequences(test\_sequences, maxlen=max\_len)

## 7. Testing the model

print('Test set\n Loss: {:0.3f}\n Accuracy: {:0.3f}'.format(accr[0],accr[1]))

Test set Loss: 0.262 Accuracy: 0.977